Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное

учреждение высшего образования

Московский технический университет связи и информатики

Кафедра «Системного программирования»

Лабораторная работа №2. Часть 1.

по дисциплине

«Операционные системы»

Выполнила: студентка группы БСТ2104

Первухина А. А.

Проверила: Алексанян Диана Ашотовна

Москва 2023

**Оглавление**

[Цели работы: 3](#_Toc146566715)

[Задание 1.1. 3](#_Toc146566716)

[Задание 1.2. 3](#_Toc146566717)

[Выполнение задания 1.1. 4](#_Toc146566718)

[Выполнение задания 1.2. 9](#_Toc146566719)

[Вывод 11](#_Toc146566720)

Цели работы: изучить архитектуру и основные компоненты ядра Linux, включая механизм системных вызовов. Изучить возможности утилиты strace для отладки и анализа работы программ на уровне системных вызовов. Приобрести практический навык создания и загрузки модуля в ядро Linux.

Задание 1.1. Анализ системных вызовов при выполнении команд с помощью утилиты strace

1. Убедитесь, что strace установлена, запустив ее с параметром -V: strace -V. Если утилита отсутствует, установите ее: sudo apt install strace.

2. Ознакомьтесь со справкой об использовании утилиты strace: man strace. 3. Запустите strace для команды из таблицы

Обратите внимание, что в некоторых случаях потребуется задать аргументы. На основании полученных результатов заполните таблицу 2 для 5-7 различных системных вызовов

4. Перенаправьте вывод strace в файл log в вашей домашней директории. 5. Получите статистику выполненных системных вызовов.

6. Выполните трассировку системных вызовов для произвольного работающего процесса, подключившись к нему по PID.

Задание 1.2. Сборка и загрузка модуля в ядро Linux

1. Установите необходимые пакеты: apt-get install gcc make linux-headers-$(uname -r)

2. Создайте файл модуля

3. Создайте Makefile

4. Соберите модуль и установите его с помощью insmod

5. Выгрузите модуль с помощью команды rmmod и включите снимок экрана вывода в отчет.

Выполнение задания 1.1.

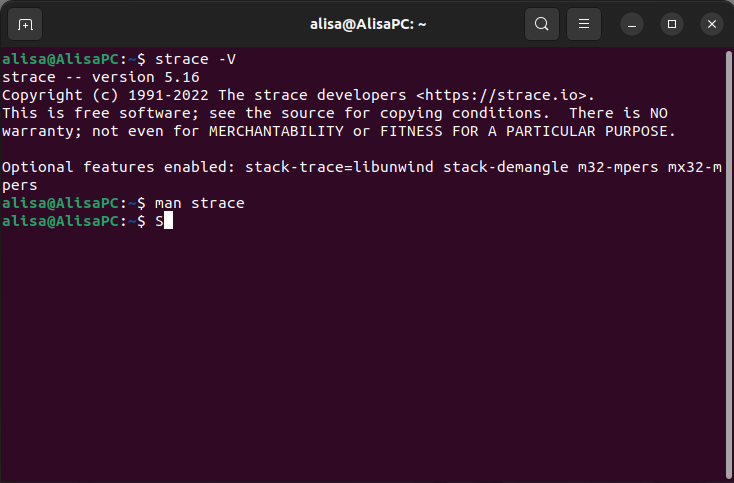
****Проверим наличие утилиты strace и ознакомимся со справкой об использовании утилиты.

Рисунок 1 – проверка наличия утилиты и вызов справки

Запустим strace для команды hostname(в соответствии с вариантом).

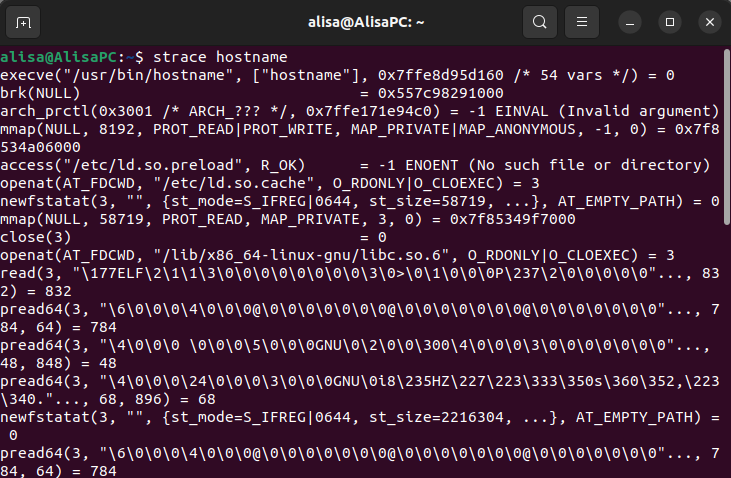


Рисунок 2 – результат работы утилиты strace

На основании полученных результатов заполним таблицу для 5-7 различных системных вызовов.

Таблица 2. Результаты анализа

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Системный  Вызов | Описание вызова | Входные параметры | Время выполнения, мкс | Возвращаемое значение |
| 1 | Execve() | Используется для выполнения новой команды в текущем процессе | **"/usr/bin/hostname" –** исполняемый файл, который должен быть запущен  **["hostname"] –** массив аргументов, передаваемых новой программе | real 9 ms  user 2 ms  sys 4 ms | **0** – успешное выполнение |
| 2 | Access | Используется для проверки наличия прав доступа к файлу/директории | **"/etc/ld.so.preload"** – Путь к файлу/директории  **R\_OK** – флаг. Вызов должен проверить, есть ли права на чтение | **-1** **ENOENT** расшифровывается как "No such file or directory" |
| 3 | Openat | Используется для открытия файла/директории | **AT\_FDCWD** - базовый дескриптор файла (в данном случае, текущий рабочий каталог).  **"/etc/ld.so.cache"** путь к файлу, который будет открыт  **O\_RDONLY|O\_CLOEXEC** флаги для открытия файла | **= 3**  файл был успешно открыт, и дескриптор файла (3) был назначен данному файлу |
| 4 | Mmap | Используется для создания отображения в памяти (memory mapping) / резервирования адресного пространства в адресном пространстве процесса. | **NULL** предпочтительное местоположение отображения в памяти  **8192** размер отображения в байтах  **PROT\_READ|PROT\_WRITE** права доступа к отображению  **MAP\_PRIVATE|MAP\_ANONYMOUS** тип отображения  **-1** файловый дескриптор  **0** смещение в файле | **= 0x7f7f87df3000**  адрес, по которому было создано отображение в памяти |
| 5 | Mprotect | Используется для изменения прав доступа к участкам памяти в процессе | **0x7f7f87c15000** адрес начала участка памяти, для которого изменяются права доступа.  **16384** размер участка памяти в байтах, для которого изменяются права доступа  **PROT\_READ** новые права доступа для участка памяти | **= 0** успешное выполнение |

В strace часто время выполнения не выводится напрямую, но мы можем измерить его, добавив time.

Использовав команду time strace hostname, получим общее время выполнения.

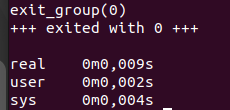


Рисунок 3 – время работы утилиты

**Real** – это общее реальное время выполнения команды **strace hostname**.

**User** – это количество времени, которое процессор был занят выполнением пользовательского кода (когда команды **hostname**).

**Sys** – это количество времени, которое процессор был занят выполнением системных вызовов (включая **strace**)

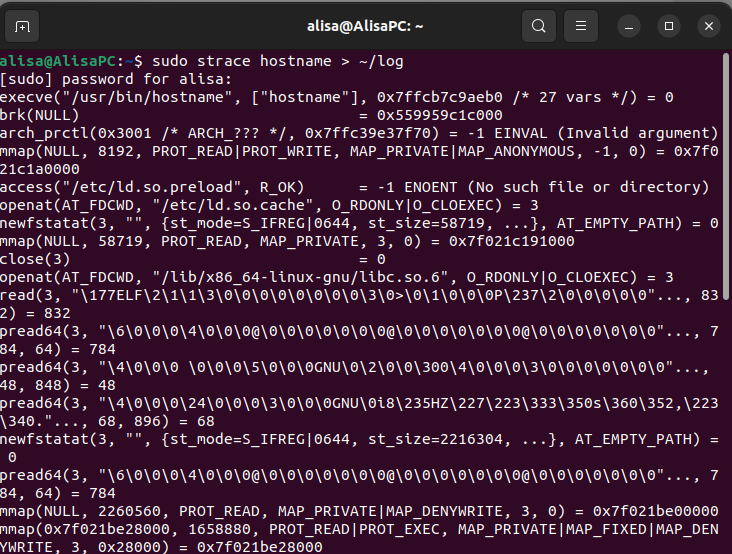
Перенаправим вывод strace в файл log в вашей домашней директории.

Рисунок 4 – вывод в log

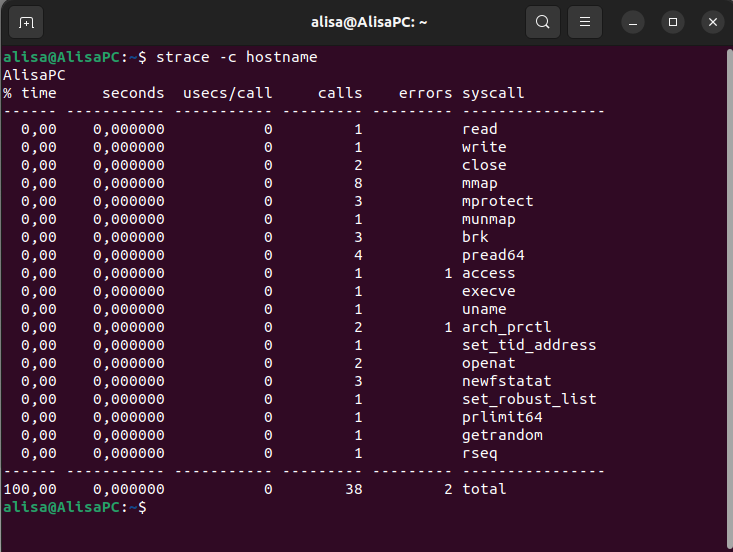
Получим статистику выполненных системных вызовов.

Рисунок 5 – статистика системных вызовов

Команда **hostname** выполнила 1 системный вызов ‘read’, 2 системных вызова ‘close’, 8 системных вызовов ‘mmap’ и так далее.

Выполним трассировку системных вызовов для произвольного работающего процесса, подключившись к нему по PID

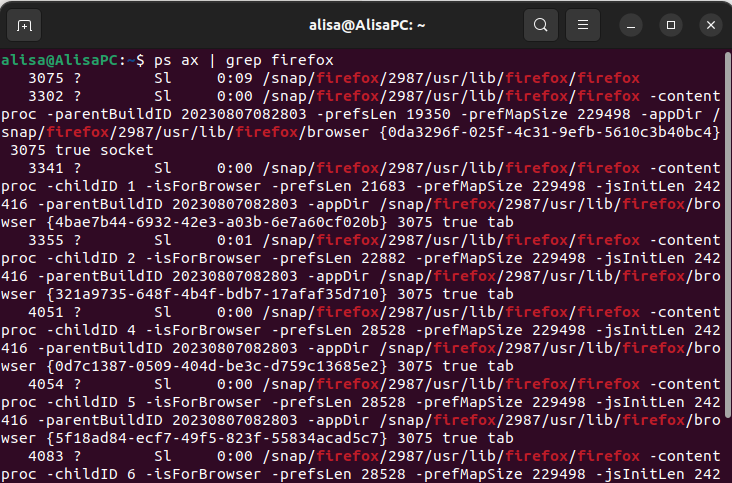
****Запустим браузер FireFox и определим PID, которые он использует.

Рисунок 6 – поиск PID в FireFox

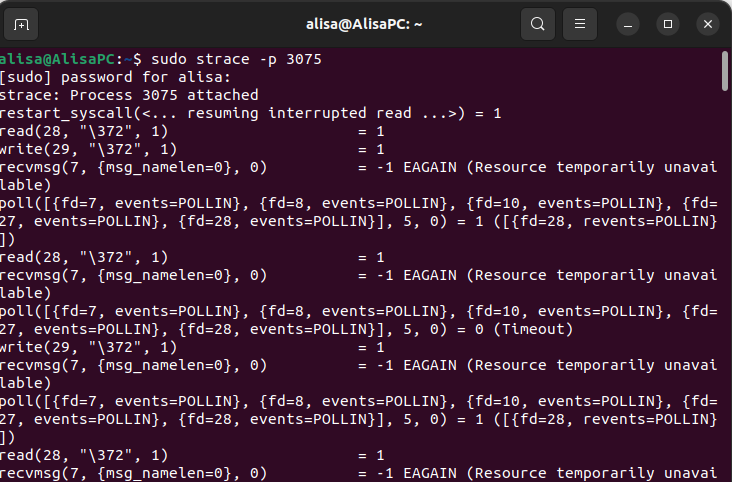
Выполним трассировку процесса с PID 3075

Рисунок 7 – трассировка процесса 3075

Выполнение задания 1.2.

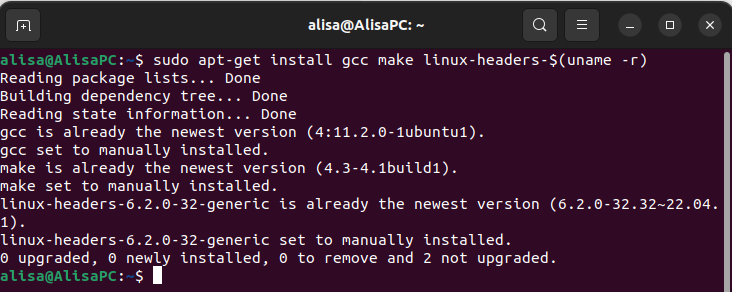
Установим необходимые пакеты

Рисунок 8 – установка пакетов

Создадим необходимые файлы.

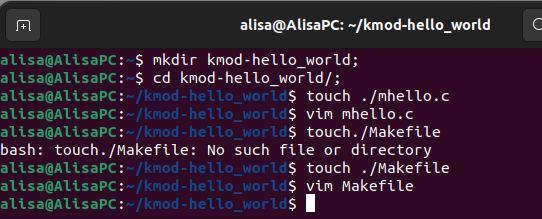


Рисунок 9 – создание и настройка файлов

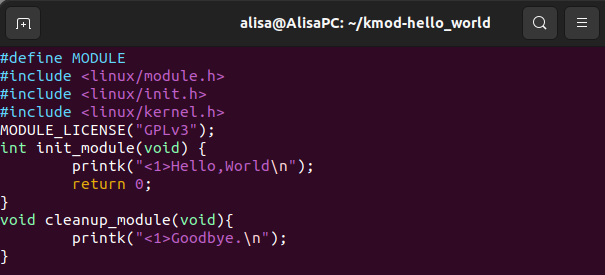
Создадим файл модуля с указанным содержимым

Рисунок 10 – файл модуля

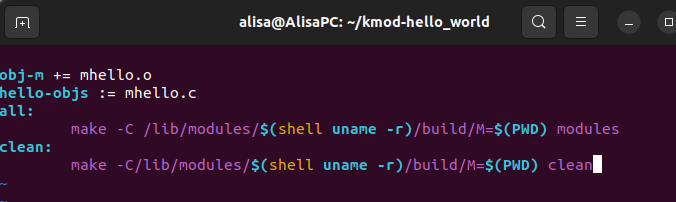
Создадим файл Makefile с указанным содержимым.

Рисунок 11 – файл Makefile

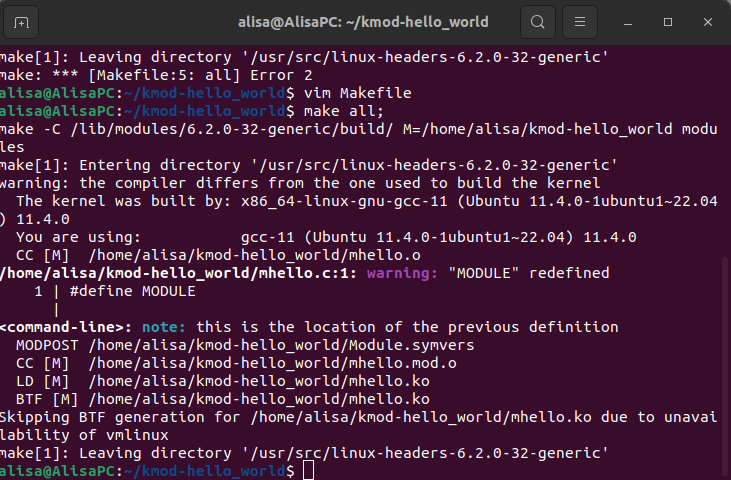
Соберём модуль.

Рисунок 12 – сборка модуля

Установим модуль и проверим это, через команду **sudo dmesg**.

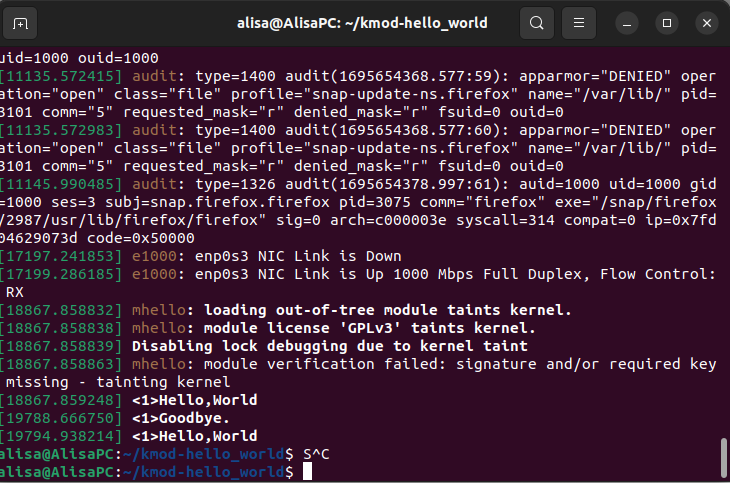


Рисунок 13 – отображение модуля в системных журналах сообщений

Выгрузим модуль и посмотрим журнал системных сообщений.

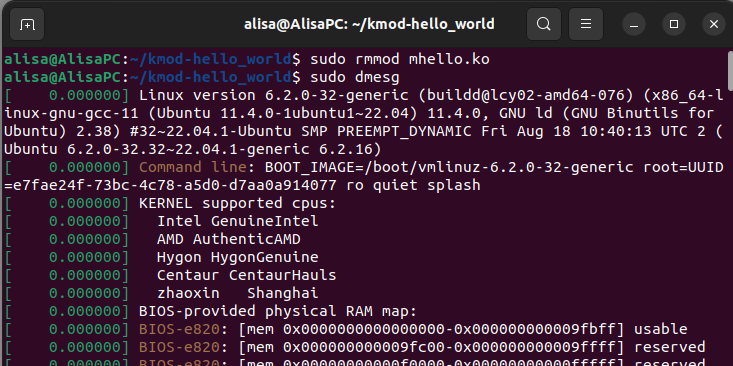


Рисунок 14 – выгрузка модуля

Вывод

Я изучила архитектуру и основные компоненты ядра Linux, включая механизм системных вызовов. Изучила возможности утилиты strace для отладки и анализа работы программ на уровне системных вызовов. Приобрела практический навык создания и загрузки модуля в ядро Linux.